(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
15 February 2001 (15.02.2001)

PCT

(10) International Publication Number
WO 01/11818 A2

(51) International Patent Classification[7]: H04L 9/00

(21) International Application Number: PCT/US00/21589

(22) International Filing Date: 7 August 2000 (07.08.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/371,147 9 August 1999 (09.08.1999) US

(71) Applicant: **QUALCOMM INCORPORATED** [US/US]; 5775 Morehouse Drive, San Diego, CA 92121-1714 (US).

(72) Inventors: **ROSE, Gregory, G.**; 6 Kingston Avenue, Mortlake, NSW 2137 (AU). **BENDER, Paul, E.**; 2879 Angell Avenue, San Diego, CA 92122 (US). **QUICK, Roy, Franklin, Jr.**; 4502 Del Monte Avenue, San Diego, CA 92107 (US). **WOLF, Jack, K.**; 8529 Prestwick Drive, La Jolla, CA 92037 (US).

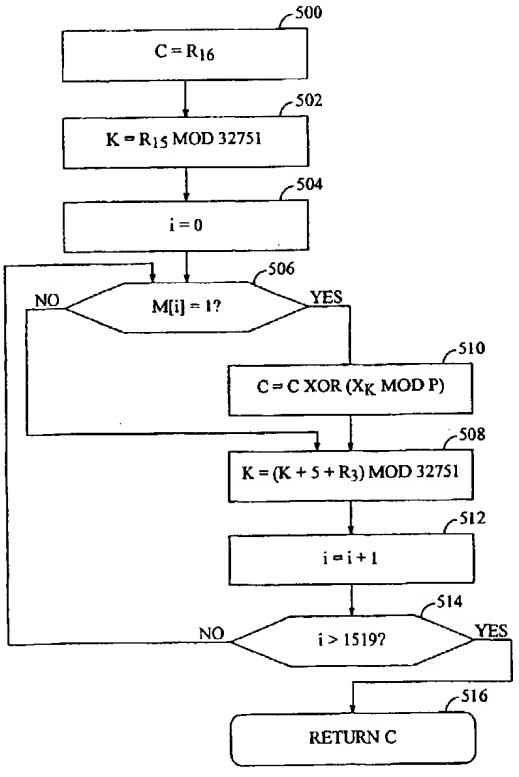
(74) Agents: **WADSWORTH, Philip, R.** et al.; Qualcomm Incorporated, 5775 Morehouse Drive, San Diego, CA 92121-1714 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: METHOD AND APPARATUS FOR GENERATING A MESSAGE AUTHENTICATION CODE



(57) Abstract: A method for generating a message authentication code (MAC) includes the steps of distributing the bits of a message into a larger message in accordance with a pseudorandom number distribution format. The cyclic redundancy check (CRC) bits of the larger message are computed and used as the MAC for the message. The larger message need not be created. The remainder modulo the CRC polynomial of a polynomial x^i, where i is the intended bit position in the "larger message", is calculated. An exclusive-OR(XOR) operation is performed, bit by bit, on the CRC and the calculated remainder to derive the new CRC.

WO 01/11818 A2

Published:

— *Without international search report and to be republished upon receipt of that report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

METHOD AND APPARATUS FOR GENERATING A MESSAGE AUTHENTICATION CODE

BACKGROUND OF THE INVENTION

I. Field of the Invention

The present invention pertains generally to the field of communications, and more particularly to generation of message authentication codes.

II. Background

A message authentication code (MAC) is a cryptographically derived item that may be appended to a particular message in order to verify that the message originated from a particular party and was not altered by any other party. It stands to reason that MACs find use in many fields of telecommunications. An exemplary field is wireless communications.

The field of wireless communications has many applications including, e.g., cordless telephones, paging, wireless local loops, wireless data applications such as personal digital assistants (PDAs), wireless telephony such as cellular and PCS telephone systems, mobile Internet Protocol (IP) telephony, and satellite communication systems. A particularly important application is wireless telephony for mobile subscribers.

Various over-the-air interfaces have been developed for wireless communication systems including, e.g., frequency division multiple access (FDMA), time division multiple access (TDMA), and code division multiple access (CDMA). In connection therewith, various domestic and international standards have been established including, e.g., Advanced Mobile Phone Service (AMPS), Global System for Mobile Communications (GSM), and Interim Standard 95 (IS-95).

An exemplary wireless telephony communication system is a code division multiple access (CDMA) system. The IS-95 standard and its derivatives, IS-95A, ANSI J-STD-008, IS-95B, proposed third generation standards IS-95C and IS-2000, proposed high-data-rate CDMA standards

exclusively for data, etc. (referred to collectively herein as IS-95), are promulgated by the Telecommunication Industry Association (TIA) and other well known standards bodies to specify the use of a CDMA over-the-air interface for cellular or PCS telephony communication systems. Exemplary wireless communication systems configured substantially in accordance with the use of the IS-95 standard are described in U.S. Patent Nos. 5,103,459 and 4,901,307, which are assigned to the assignee of the present invention and fully incorporated herein by reference.

In a typical communication, the MAC m is the output of a function computed with the message M (of length L_M) and a shared secret key K known only by the message originator and recipient as the inputs to the function. If the particular function chosen is secure, then an active attacker who can intercept and potentially modify the messages sent can neither discover the key K nor create messages that will be accepted by the recipient as valid with a reasonable probability. If the MAC has a length L_m (in bits), then the attacker could always simply guess a value for m for a desired message, and with a probability of $1/2^{Lm}$ the guess will be correct. Hence, any guarantee of security for a MAC is, by nature, probabilistic. MACs in general provide no better guarantee of detecting an error, whether deliberately or randomly introduced, than this probability. In particular, a single-bit error in a message generally has the same chance of matching the MAC attached to the message as would any other substitution. While this probability is small, it is still significant.

A cyclic redundancy check (CRC) is one example of a well-known error detection-and-correction code (ECC). ECCs are used in many applications in which data transmissions may be corrupted, and when the receiver wants to be able to detect the most common corruptions and correct them. CRCs are efficient to compute, and have useful error detection properties. If the received message M' has an error in any small number of bits, the CRC will guarantee to detect that there has been an error, and in fact will indicate (on the assumption that the error was the smallest possible) which bits were in error. This enables the correction of the error. CRCs are calculated by considering the bits of the

message to be the coefficients of a polynomial and calculating the remainder when the polynomial is divided by a polynomial P of degree L. Careful choice of the polynomial P gives the desired error detection and correction properties. While CRCs are good for detecting the kinds of random errors introduced during transmission, they are useless for preventing any kind of active attack because the attacker can easily calculate the effect on the CRC of any modification of a message. The attacker can also modify the CRC accordingly. This remains true even if there is some secret information incorporated in the calculation of the CRC but not transmitted with the message.

It would be desirable to provide a combined MAC/CRC, i.e., a code that includes a guarantee that small, random errors that would have been detected by the CRC will be detected by the mismatch of the message and its MAC. It would further be advantageous for the code to simultaneously guarantee that an active attacker who does not know the secret information K will not be able to discover K, or "forge" a message with a probability of better than $1/(2^K-1)$. (It should be noted that, as would be understood by those of skill in the art, the extra "-1" in the probability value comes from the fact that the attacker knows, in this case, that the original MAC will not work for a message that is mostly the same as the one the attacker is modifying, so the attacker will choose one of the other possible MACs at random instead.) Thus, there is a need for a method of generating a MAC with guaranteed corruption detection properties.

SUMMARY OF THE INVENTION

The present invention is directed to a method of generating a MAC with guaranteed corruption detection properties. Accordingly, in one aspect of the invention, a method of generating a message authentication code advantageously includes the steps of pseudorandomly distributing in a key-dependent manner a first plurality of message bits into a second plurality of bits; generating a third plurality of bits that comprises a cyclic redundancy check of the second plurality of bits; and transmitting the first plurality of message bits and a message authentication code that comprises the third plurality of bits.

In another aspect of the invention, a generator configured to generate a message authentication code advantageously includes means for pseudorandomly distributing in a key-dependent manner a first plurality of message bits into a second plurality of bits; means for generating a third plurality of bits that comprises a cyclic redundancy check of the second plurality of bits; and means for transmitting the first plurality of message bits and a message authentication code that comprises the third plurality of bits.

In another aspect of the invention, a generator configured to generate a message authentication code advantageously includes a processor configured to pseudorandomly distribute in a key-dependent manner a first plurality of message bits into a second plurality of bits; a generator coupled to the distributor and configured to generate a third plurality of bits that comprises a cyclic redundancy check of the second plurality of bits; and a transmitter coupled to the generator and configured to transmit the first plurality of message bits and a message authentication code that comprises the third plurality of bits.

In an embodiment of the invention, a method of pseudorandomly distributing bits of a message advantageously includes the steps of calculating a remainder modulo P of a polynomial x^i, where i is an intended message bit location and P is a cyclic redundancy check polynomial; and, for each bit of the message that is equal to one, performing an exclusive-OR operation, bit by bit, of the CRC bits and the calculated remainder.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of a cellular telephone system.

FIG. 2 is a block diagram of a processor and associated memory elements used to generate a message and an associated message authentication code (MAC).

FIG. 3 is a block diagram of a generator for generating a message and an associated MAC.

FIG. 4 is a schematic diagram of registers that could be used in the generator of FIG. 3 to employ a keyed pseudorandom number (PRN) message bit distribution technique.

FIG. 5 is a schematic diagram of a cyclic redundancy check (CRC) generator that could be used in the generator of FIG. 3.

FIG. 6 is a flow chart illustrating method steps performed by a generator such as the generator of FIG. 3 to generate a MAC for a message.

FIG. 7 is a flow chart illustrating method steps performed by a generator such as the generator of FIG. 3 to generate a MAC for a message.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The exemplary embodiments described hereinbelow reside in a wireless telephony communication system configured to employ a CDMA over-the-air interface. Nevertheless, it would be understood by those skilled in the art that a MAC generation method and apparatus embodying features of the instant invention may reside in any of various communication systems employing a wide range of technologies known to those of skill in the art.

As illustrated in FIG. 1, a CDMA wireless telephone system generally includes a plurality of mobile subscriber units 10, a plurality of base stations 12, base station controllers (BSCs) 14, and a mobile switching center (MSC) 16. The MSC 16 is configured to interface with a conventional public switch telephone network (PSTN) 18. The MSC 16 is also configured to interface with the BSCs 14. The BSCs 14 are coupled to the base stations 12 via backhaul lines. The backhaul lines may be configured to support any of several known interfaces including, e.g., E1/T1, ATM, IP, PPP, Frame Relay, HDSL, ADSL, or xDSL. It is understood that there may be more than two BSCs 14 in the system. Each base station 12 advantageously includes at least one sector (not shown), each sector comprising an omnidirectional antenna or an antenna pointed in a particular direction radially away from the base station 12. Alternatively, each sector may comprise two antennas for diversity reception. Each base station 12 may

advantageously be designed to support a plurality of frequency assignments. The intersection of a sector and a frequency assignment may be referred to as a CDMA channel. The base stations 12 may also be known as base station transceiver subsystems (BTSs) 12. Alternatively, "base station" may be used in the industry to refer collectively to a BSC 14 and one or more BTSs 12. The BTSs 12 may also be denoted "cell sites" 12. Alternatively, individual sectors of a given BTS 12 may be referred to as cell sites. The mobile subscriber units 10 are typically cellular or PCS telephones 10. The system is advantageously configured for use in accordance with the IS-95 standard.

During typical operation of the cellular telephone system, the base stations 12 receive sets of reverse link signals from sets of mobile units 10. The mobile units 10 are conducting telephone calls or other communications. Each reverse link signal received by a given base station 12 is processed within that base station 12. The resulting data is forwarded to the BSCs 14. The BSCs 14 provides call resource allocation and mobility management functionality including the orchestration of soft handoffs between base stations 12. The BSCs 14 also routes the received data to the MSC 16, which provides additional routing services for interface with the PSTN 18. Similarly, the PSTN 18 interfaces with the MSC 16, and the MSC 16 interfaces with the BSCs 14, which in turn control the base stations 12 to transmit sets of forward link signals to sets of mobile units 10.

In accordance with one embodiment, as shown in FIG. 2, a mechanism 100 for generating a message including a MAC includes a processor 102, a software module 104, and a storage medium 106. The processor 102 is advantageously a microprocessor or a special-purpose processor such as a digital signal processor (DSP), but may in the alternative be any conventional form of processor, controller, microcontroller, or state machine. The processor 102 is coupled to the software module 104, which is advantageously implemented as RAM memory holding software instructions that direct the operation of the processor 102. The software instructions can comprise a software program or a set of microcodes. The RAM memory 104 may be on-

board RAM, or the processor 102 and the RAM memory 104 could reside in an ASIC. In an alternate embodiment, firmware instructions are substituted for the software module 104. The storage medium 106 is coupled to the processor 102, and is advantageously implemented as a combination of RAM memory and any form of conventional nonvolatile memory such as, e.g., ROM memory. The storage medium 106 is used to implement a linear feedback shift register (LFSR) to generate the MAC, as described hereinbelow, and to store precomputed tables and instructions. For example, the instructions and tables are stored in the ROM memory component while the register is stored in the RAM memory component. In the alternative, the storage medium 106 could be implemented as either a disk memory or a flash memory that is accessible by the processor 102. In the alternative, the storage medium 106 may be implemented as registers. The mechanism 100 may reside in any conventional communications device such as, e.g., the mobile subscriber units 10 or the base stations 12 in the CDMA wireless telephone system of FIG. 1.

In one embodiment a generator 200 for generating a message and an associated MAC includes a keyed pseudorandom number (PRN) distributor 202, a cyclic redundancy check (CRC) generator 204, a modulator 206, and a transmitter 208, as shown in FIG. 3. Message bits of a message, *M*, are provided to the keyed PRN distributor 202. The keyed PRN distributor 202 distributes the message bits pseudorandomly in a key-dependent manner, as described in detail below, into a bit sequence. The bit sequence including the distributed message bits is provided to the CRC generator 204. The CRC generator 204 computes the CRC of the bit sequence including the distributed bits in accordance with any conventional method for CRC calculation as known to those of skill in the art.

The CRC generator 204 generates the CRC bits, which will be used as a MAC for the message bits. The MAC bits and the message bits are provided to the modulator 206. The modulator 206 modulates the received bits for transmission on a communication channel. The modulation scheme varies with the communication system and the type of communication channel being used.

In one embodiment the modulation scheme is a CDMA scheme and the communication system is the wireless telephone system of FIG. 1. The modulator 206 provides a modulated message and MAC signal to the transmitter 208. The transmitter 208 transmits the modulated message and MAC signal on the communication channel.

In accordance with the embodiment described with reference to FIG. 3, the CRC of a larger "message" (the bit sequence) is calculated and transmitted as a MAC in which the bits of the original message, *M*, have been distributed in a key-dependent manner that cannot be predicted by an attacker. Thus, small random errors can be detected and corrected by the normal CRC mechanism, while active attacks (i.e., intentional modification of an otherwise legitimate message) have only a limited probability of success.

Advantageously, the method of distributing bits of the original message, *M*, through the larger message varies from message to message. The variation in the distribution method prevents an attacker from gathering messages and thereby gradually increasing the probability of a successful attack. Because the MAC must guarantee to detect small errors, if two similar messages are transmitted (with, by definition, different MACs) then the MAC on another message which is similar to, but different from, both of the two similar messages must also be different, so that the attacker can choose at random from only 2^L-2 possible MACs, etc. Advantageously, information denoted "salt", *S*, is used to associate information about a particular message, such as, e.g., the time at which the message is transmitted, or a sequence number, with the manner in which the message bits are distributed. This is similar to the fact that a stream cipher should never generate the same stream of output for two different messages, or for two different segments of a single message.

Accordingly, it is necessary to distribute the bits from the original message, *M*, in the larger "message" in a manner that is unpredictable to the attacker and does not help the attacker discover information about the shared secret key, *K*. Thus, in accordance with the embodiment described with reference to FIG. 3, uniformly distributed PRNs are generated to control the

distribution of message bits within the larger message. The uniformly distributed PRNs are advantageously derived from the shared secret key, *K*, and the salt, *S*, in a manner that is unpredictable to the attacker.

In an exemplary embodiment, the output of a stream cipher called SOBER is used as a source of the PRNs. The SOBER stream cipher is described in U.S. Application Serial No. 09/246,366, entitled METHOD AND APPARATUS FOR GENERATING ENCRYPTION STREAM CIPHERS, filed February 8, 1999, and assigned to the assignee of the present invention. Stream ciphers are pseudorandomly generated bit streams that are exclusive-OR'ed (XORed), bit by bit, with respective bits of a message to be transmitted, thereby generating an encrypted message. When the encrypted message is received, it is XORed with the same stream cipher in order to produce the original message. In alternate embodiments, other forms of PRN generator may be substituted for the stream cipher. In particular, PRN generators that provide less security than a stream cipher could be used in place of the stream cipher.

In one embodiment, as illustrated in FIG. 4, the bits of a message *M* 300 are distributed under the control of a keyed PRN generator (not shown) by starting at some offset in a larger message 302 and placing bits sequentially, skipping an unpredictable number of positions in the larger message 302 between bits. When, or if, the end of the larger message 302 is encountered, the distribution position starts again at the beginning of the larger message 302. The maximum number of positions skipped between bits of the larger message 302 is advantageously determined so that the distribution position does not completely wrap around, i.e., so that the initial position in the larger message 302 is not reached or passed. This ensures that no two bits of the message *M* 300 will be distributed to the same position in the larger message 302. If two bits were distributed to the same position in the larger message 302, a simultaneous change to both of these bits would cancel out, and not be detected, which would not achieve the goal. To allow for the worst case, wherein every gap between distributed bits is maximum length, the maximum gap length must be limited. However the average gap length is only half as

long as the maximum gap length. Therefore, on average, the message bits are distributed only within half of the larger message. This distribution technique advantageously provides ample security and relative ease of implementation.

In another embodiment the message bits 300 are distributed by dividing the larger message 302 into approximately equal sized blocks, starting at a random offset and wrapping around, and placing one bit at a random position within each block. This distribution technique has the desirable property that the bits will be well distributed throughout the larger message.

The bits of the larger message 302 are provided to a CRC generator 304, which calculates the CRC of the received bits. The CRC bits 306 are used as a MAC 306 for the message *M* 300. It should be pointed out that, as illustrated in connection with other embodiments described below with reference to FIG. 6, it is not necessary to actually create the larger message 302 to achieve the desired effect.

In an exemplary embodiment, the length L_m of the CRC/MAC is sixteen bits. The largest message for which the guarantees for error detection apply is 2^{16} bits including the CRC itself. (If $(1+x)$ multiplied by a primitive polynomial were used as the generator for the CRC, the largest message length would be 2^{15}-1 bits.) The length L_M of the input message *M* 300 is advantageously kept significantly smaller than 2^{15}-16 bits. If the larger message 302 is divided into roughly equal sized blocks, in accordance with the second embodiment described above, the length L_M of the input message *M* 300 is advantageously restricted to less than half of 2^{15}-16 bits. If bit positions are skipped within the larger message 302, in accordance with the first embodiment described above, the length L_M of the input message *M* 300 is advantageously restricted to less than one-quarter of 2^{15}-16 bits.

After the first bit of the message *M* 300 has been placed, the remaining L_M-1 bits are placed in the remaining 2^{15}-17 positions. If the length L_M of the input message, *M*, is 1520 bits, the maximum distribution spacing or the size of the block would be the greatest integer less than 32751/1519, or twenty-one bits. However the PRN generator can be viewed as producing a stream of bits.

Accordingly, it may advantageously be more efficient to use a number that is a power of two, instead of using twenty-one bits. If a power of two is used as maximum distribution spacing or the block size, random numbers in the range zero to fifteen could be generated by taking four bits of output from the PRN generator. The random numbers are then advantageously adjusted to be in the range five to twenty. It would be understood by those of skill in the art that the average spacing between bits or the size of the blocks must be at least two. Hence, the optimal values for the spacing or the block size are the powers of two between 2^2 and 2^4. Placement of each bit of the message M 300 therefore requires two to four bits of PRN output.

As described further below, it may be advantageous to spread the bits further apart, even if the uncertainty in the position of the bits is limited by accepting less output from the PRN generator. A minimum spread size or minimum block size, whether constant or variable, could be used.

In another embodiment, message bits are distributed by using a keyed permutation function such as a block cipher to derive new bit positions from the original bit positions. In a particular embodiment, a block cipher with a block size of fourteen bits is employed in conjunction with a random offset position in the larger message 302. It should be noted that the necessity for using a different permutation for each message requires the block cipher to use a different key for each message 300. While this is theoretically possible, it would be inefficient in practice.

In one embodiment, illustrated in FIG. 5, a CRC generator 400 is implemented as a register that includes sixteen one-bit storage elements 402a-g (only storage elements 402 are shown for the purpose of simplification), three modulo-2 adders 404, 406, 408, and three switches 410, 412, 414. In an alternate embodiment a CRC generator is implemented with a microprocessor running a set of software instructions, advantageously contained in RAM memory, and accessing lookup tables (LUTs), advantageously contained in ROM memory or flash memory, as described above with reference to FIG. 2.

In the CRC generator 400, input message bits are provided to the switch 410, which may either be set to receive the input message bits or be set to receive a digital value of one. The switch 412 may either be set to receive a digital value of zero or be set to receive a value from the modulo-2 adder 408. The switch 414 may either be set to receive a value from the switch 410 or to receive a value from the switch 412 and the modulo-2 adder 408. An output CRC is taken from the switch 414. The modulo-2 adder 404 is located between the fifth one-bit storage element 402c and the sixth one-bit storage element 402d. The modulo-2 adder 406 is located between the twelfth one-bit storage element 402e and the thirteenth one-bit storage element 402f. The modulo-2 adder 408 is located after the sixteenth one-bit storage element 402h and is configured to receive a value from the switch 410. The generator polynomial $g(x)$ for the CRC is equal to $x^{16} + x^{12} + x^5 + 1$, as defined by the placement of the modulo-2 adders 404, 406, 408.

In operation the switches 410, 412, 414 are set initially in the "up" positions (as shown in the drawing). The register is clocked k times, where k is defined as the length of the input message plus eight bits. The register is a shift register such that with each clock cycle the bits each move one storage element to the right (as shown in the drawing). The switches 410, 412, 414 are then set to the "down" position (as shown in the drawing). The register is then clocked an additional sixteen times. The sixteen additional output bits comprise the CRC field for the message. The bits are transmitted in the order at which they appear at the output of the CRC generator 400.

The input message bits constitute a "larger message." The CRC field for the larger message serves as a MAC for the message M. Advantageously then, the MAC includes the inherent security benefits of a CRC. Because the MAC calculated is actually a CRC, the guarantees about error detection and correction that apply to CRCs apply to the MAC, with the exception of guarantees applying to "burst errors." Burst error guarantees are violated because consecutive bits are separated during the calculation and therefore no longer form a "burst."

There are essentially two kinds of attacks against MACs. The first kind of attack against a MAC is theft of service. The attacker tries to create a message with a valid MAC, having observed other valid messages. Alternatively, the attacker tries to deduce the secret key, *K*, allowing the attacker to create forged messages at will. The second kind of attack against a MAC is chosen message. The attacker tries to craft specific messages and somehow contrive for the system to calculate the valid MAC, with a view to recovery of the secret key, *K*.

An attacker wishing to make a single bit alteration to a message with a valid MAC would need to be able to calculate the corresponding alteration to the CRC; this in turn requires being able to predict the position of the bit in the expanded message. Since there are just under 2^{L-1} possible positions of the bit, the attacker can do only slightly better than just guessing the MAC.

One possible attack against MACs of any kind is a so-called chosen plaintext attack. In this type of attack, the attacker can somehow arrange to have a message transmitted, accompanied with a valid MAC, such that the message has a particular content chosen by the attacker. For example, the attacker might send electronic mail to the recipient, which would eventually be routed through the transmission channel and have a valid MAC computed and attached. The attacker might make such a message consisting of a single "1" bit followed by all "0" bits. By observing the calculated CRC, the attacker would be able to infer the initial position, and hence some of the output of the PRN generator. This is undesirable, as it may lead to a method of predicting future outputs. Accordingly, in one embodiment, one other bit of the larger message is set to one. The other bit is chosen at a random position.

The attacker has the alternative of sending a message comprising all zeroes. The observed CRC gives the attacker L_m bits of output from the PRN generator, which needs to be secure enough to prevent recovery of the secret key information, *K*, given this disclosure.

If the attacker tries to make a multi-bit alteration to the message in addition to predicting the starting position of the bits, the attacker must also

determine the dispersion of the bits. This would allow a smaller probability of success than if done for only a single bit.

There remains one modification attack that is applicable. If the attacker can make a change that will be spread out into a pattern that is a multiple of the CRC polynomial, there will be no effect on the calculated MAC. This effectively nullifies the unpredictability of the initial position of the bits. If it is desirable to use a small number of stream cipher bits, it is advantageous to choose the method of spreading the bits with consideration for the particular CRC polynomial, so as to make this type of attack impossible for small multiples of the polynomial and statistically difficult for large multiples of the polynomial.

In one embodiment a CRC computation is performed on a "larger message" to generate a MAC for an input message, *M*, without actually creating the larger message. For each bit of the input message, *M*, to be distributed in the fictitious "larger message," it is necessary to calculate the remainder modulo *P* of the polynomial x^i, where *i* is the intended position of the message bit in the larger message. Because the CRC is linear, the remainder calculation can be added by polynomial addition, i.e., an exclusive-or (XOR) operation, to the existing CRC if the particular bit of the message is a one. The CRC of the all-zero message is zero, so by applying the XOR polynomial addition technique to each of the bits in the input message, *M*, in turn, the CRC can be calculated without ever creating the larger message, and also by performing only L_M calculations. Similarly, the initial seeding of the input message, *M*, with a non-zero bit is equivalent to choosing a random starting CRC for the calculation.

Calculating $x^i \bmod P$, where *i* is in the range 0 .. 32767 (assuming that L_m is sixteen) can be done in a variety of ways. In a particular embodiment, a pair of lookup tables (LUTs) is used. It should be pointed out that a single LUT that gave the CRC corresponding to each possible value of *i* in the above equation would suffice, but would require 2^{16} 16-bit table entries. Instead, *i* is advantageously expressed in the form $256hi + lo$, where *hi* and *lo* are the high-order and low-order eight bits of *i*, respectively. Two LUTs are precalculated to

give the results of the CRC calculation x^{256hi} *mod P* and x^{lo} *mod P*, respectively, so that performing an XOR operation on each entry in the first LUT with the corresponding entry in the second LUT is equivalent to calculating the CRC. Accordingly, two LUTs are required, each LUT having 256 16-bit entries, and two table lookups are performed per nonzero bit of the input message, *M*, to calculate the CRC.

It may also be advantageous in a software implementation to note that as the bits are being distributed through the larger message, *hi* does not change very rapidly, so that repeated lookups of the same value would cancel each other out. All that would actually count in the final result would be whether each value of *hi* was used an odd number of times.

In one embodiment a PRN generator generates a MAC for an input message, *M*, as the CRC of a fictitious "larger message" without creating the larger message in accordance with the algorithm steps illustrated in the flow chart of FIG. 6. In this particular embodiment, L_m is sixteen bits and L_M is 1520 bits. The CRC polynomial, *P*, is the 16-bit CRC-CCITT polynomial $P(x) = x^{16} + x^{12} + x^5 + 1$. Specific LUTs for looking up the CRC values may be readily calculated, as understood by those of skill in the art. The notation R_n, where n is an integer, is used below to indicate that the next *n* bits of output from the PRN generator (not shown) should be taken. In this particular embodiment, a constant minimum spacing of five bits is employed, and three bits of PRN output are used, giving a spacing between uniformly distributed bits of between five and twelve bits. The variable *C* denotes the accumulator of the eventual output MAC. The variable *K* denotes the position of the next bit to be placed in the "larger message." The variable *i* denotes the position of the bit in the input message, *M*.

In step 500 the generator sets *C* equal to R_{16}, which is equivalent to choosing a random bit position to be set to one. The next sixteen bits of generator output will form the CRC. The generator then proceeds to step 502. In step 502 the generator sets *K* equal to R_{15} modulo 32751, which is effectively the position of the first bit of the input message, *M*, to be distributed. The

generator then proceeds to step 504. In step 504 the generator sets *i* equal to zero. The generator then proceeds to step 506.

In step 506 the generator determines whether the bit *M[i]* (the input message bit currently being distributed) is set to one. If the bit *M[i]* is not set to one, the generator proceeds to step 508. If, on the other hand, the bit *M[i]* is set to one, the generator proceeds to step 510. In step 510 the generator calculates x^K modulo *P*, and performs an XOR operation, bit by bit, on *C* and x^K modulo *P*, in accordance with the polynomial addition technique described above. The new value of *C* is set equal to the bits resulting from the XOR calculation. The generator then proceeds to step 508.

In step 508 the generator computes the sum of *K*, five, and R_3 (the next three bits of generator output), modulo 32751. The result is set equal to *K*, so that *K* is updated to be the location in which the next bit of the input message is to be distributed. This calculation is performed after distribution of the current bit, *M[I]*, in step 510. The generator then proceeds to step 512. In step 512 the generator increments *i* by one. The generator then proceeds to step 514. In step 514 the generator determines whether *i* is greater than 1519. If *i* is not greater than 1519, the generator returns to step 506 to process the next input message bit, *M[i]*. If, on the other hand, *i* is greater than 1519, the generator proceeds to step 516. In step 516 the generator returns the value of *C* as the MAC for the input message.

In an alternate embodiment, the calculation performed in step 508 is to compute the sum of *K*, ten, and R_4 (the next four bits of generator output), modulo 65521. Again, the result is set equal to *K*. In accordance with this embodiment, the generator sets *K* equal to R_{16} modulo 65521 in step 502. A primitive polynomial that may be used in accordance with this embodiment is $x^{16} + x^{14} + x^{12} + x^7 + x^6 + x^5 + x^2 + x + 1$.

In another embodiment a PRN generator generates a MAC for an input message, *M*, as the CRC of a fictitious "larger message" without creating the larger message in accordance with the algorithm steps illustrated in the flow

chart of FIG. 7. In this particular embodiment, L_m is sixteen bits and L_M is 1520 bits. The CRC polynomial, *P*, is the 16-bit CRC-CCITT polynomial $P(x) = x^{16} + x^{12} + x^5 + 1$. Specific LUTs for looking up the CRC values may be readily calculated, as understood by those of skill in the art. The notation R_n, where n is an integer, is used below to indicate that the next *n* bits of output from the PRN generator (not shown) should be taken. In this particular embodiment, a constant minimum spacing of five bits is employed, and three bits of PRN output are used, giving a spacing between uniformly distributed bits of between five and twelve bits. The variable *C* denotes the accumulator of the eventual output MAC. The variable *K* denotes the position of the next bit to be placed in the "larger message." The variable *i* denotes the position of the bit in the input message, *M*.

In step 600 the generator sets *C* equal to R_{16}, which is equivalent to choosing a random bit position to be set to one. The next sixteen bits of generator output will form the CRC. The generator then proceeds to step 602. In step 602 the generator sets *K* equal to R_{15} modulo 32751, which is effectively the position of the first bit of the input message, *M*, to be distributed. The generator then proceeds to step 604. In step 604 the generator sets *i* equal to zero. The generator then proceeds to step 606.

In step 606 the generator determines whether the bit *M[i]* (the input message bit currently being distributed) is set to one. If the bit *M[i]* is not set to one, the generator proceeds to step 608. If, on the other hand, the bit *M[i]* is set to one, the generator proceeds to step 610. In step 610 the generator calculates x^K modulo *P*, and performs an XOR operation, bit by bit, on *C* and x^K modulo *P*, in accordance with the polynomial addition technique described above. The new value of *C* is set equal to the bits resulting from the XOR calculation. The generator then proceeds to step 608.

In step 608 the generator the generator increments *i* by one. The generator then proceeds to step 612. In step 612 the generator determines whether *i* is greater than 1519. If *i* is not greater than 1519, the generator proceeds to step 614. If, on the other hand, *i* is greater than 1519, the generator

proceeds to step 616. In step 616 the generator returns the value of C as the MAC for the input message. In step 614 the generator computes the sum of K, five, and R_3 (the next three bits of generator output), modulo 32751. The result is set equal to K, so that K is updated to be the location in which the next bit of the input message is to be distributed. This calculation of the new bit location is performed when a new bit is about to be distributed. After performing the calculation of step 614, the generator returns to step 606 to process the next input message bit, *M[i]*.

In an alternate embodiment, the calculation performed in step 614 is to compute the sum of K, ten, and R_4 (the next four bits of generator output), modulo 65521. Again, the result is set equal to K. In accordance with this embodiment, the generator sets K equal to R_{16} modulo 65521 in step 602. A primitive polynomial that may be used in accordance with this embodiment is $x^{16} + x^{14} + x^{12} + x^7 + x^6 + x^5 + x^2 + x + 1$.

Thus, a novel method and apparatus for generating a MAC have been described. Those of skill in the art would understand that the various illustrative logical blocks and algorithm steps described in connection with the embodiments disclosed herein may be implemented or performed with a digital signal processor (DSP), an application specific integrated circuit (ASIC), discrete gate or transistor logic, discrete hardware components such as, e.g., registers and FIFO, a processor executing a set of firmware instructions, or any conventional programmable software module and a processor. The processor may advantageously be a microprocessor, but in the alternative, the processor may be any conventional processor, controller, microcontroller, or state machine. The software module could reside in RAM memory, flash memory, registers, or any other form of writable storage medium known in the art. Those of skill would further appreciate that the data, instructions, commands, information, signals, bits, symbols, and chips that may be referenced throughout the above description are advantageously represented by voltages, currents, electromagnetic waves, magnetic fields or particles, optical fields or particles, or any combination thereof.

Preferred embodiments of the present invention have thus been shown and described. It would be apparent to one of ordinary skill in the art, however, that numerous alterations may be made to the embodiments herein disclosed without departing from the spirit or scope of the invention. Therefore, the present invention is not to be limited except in accordance with the following claims.

What is claimed is:

CLAIMS

1. A method of generating a message authentication code, comprising the steps of:

pseudorandomly distributing in a key-dependent manner a first plurality of message bits into a second plurality of bits;

generating a third plurality of bits that comprises a cyclic redundancy check of the second plurality of bits; and

transmitting the first plurality of message bits and a message authentication code that comprises the third plurality of bits.

2. The method of claim 1, wherein the generating step is performed with a microprocessor, lookup tables accessible by the microprocessor, and a set of software instructions executable by the microprocessor.

3. The method of claim 1, wherein the generating step is performed with a shift register.

4. The method of claim 1, wherein the pseudorandomly distributing step comprises the steps of:

placing a first bit of the first plurality of bits into an offset bit location in the second plurality of bits; and

skipping an unpredictable number of bit locations in the second plurality of bits from the offset bit location.

5. The method of claim 4, further comprising the step of putting a next bit of the first plurality of bits into an initial bit location of the second plurality of bits.

6. The method of claim 4, further comprising the step of determining a maximum number of bit locations skipped in the skipping step so that an initial bit location of the second plurality of bits is not reached or passed.

7. The method of claim 4, wherein the number of bit locations skipped in the skipping step is a power of two between four and sixteen, inclusive.

8. The method of claim 1, wherein the pseudorandomly distributing step comprises the steps of:

dividing the second plurality of bits into blocks of approximately uniform block size; and

placing the first plurality of bits into random bit locations within each block, with one bit being placed into each block.

9. The method of claim 8, wherein the uniform block size is a power of two between four and sixteen, inclusive.

10. The method of claim 1, wherein the pseudorandomly distributing step comprises the step of deriving new bit locations from original bit locations of the first plurality of bits in the second plurality of bits with a keyed permutation function.

11. The method of claim 1, wherein the pseudorandomly distributing step comprises the steps of:

calculating a remainder modulo P of a polynomial x^i, where i is an intended bit location in the second plurality of bits and P is a cyclic redundancy check polynomial derived from the third plurality of bits; and

for each bit of the first plurality of bits that is equal to one, performing an exclusive-OR operation, bit by bit, of the third plurality of bits and the calculated remainder.

12. A generator configured to generate a message authentication code, comprising:

means for pseudorandomly distributing in a key-dependent manner a first plurality of message bits into a second plurality of bits;

means for generating a third plurality of bits that comprises a cyclic redundancy check of the second plurality of bits; and

means for transmitting the first plurality of message bits and a message authentication code that comprises the third plurality of bits.

13. The generator of claim 12, wherein the means for generating comprises a microprocessor, lookup tables accessible by the microprocessor, and a set of software instructions executable by the microprocessor.

14. The generator of claim 12, wherein the means for generating comprises a shift register.

15. The generator of claim 12, wherein the means for pseudorandomly distributing comprises:

means for placing a first bit of the first plurality of bits into an offset bit location in the second plurality of bits; and

means for skipping an unpredictable number of bit locations in the second plurality of bits from the offset bit location.

16. The generator of claim 15, further comprising means for putting a next bit of the first plurality of bits into an initial bit location of the second plurality of bits.

17. The generator of claim 15, further comprising means for determining a maximum number of bit locations skipped so that an initial bit location of the second plurality of bits is not reached or passed.

18. The generator of claim 15, wherein the number of bit locations skipped is a power of two between four and sixteen, inclusive.

19. The generator of claim 12, wherein the means for pseudorandomly distributing comprises:

means for dividing the second plurality of bits into blocks of approximately uniform block size; and

means for placing the first plurality of bits into random bit locations within each block, with one bit being placed into each block.

20. The generator of claim 19, wherein the uniform block size is a power of two between four and sixteen, inclusive.

21. The generator of claim 12, wherein the means for pseudorandomly distributing comprises means for deriving new bit locations from original bit locations of the first plurality of bits in the second plurality of bits with a keyed permutation function.

22. The generator of claim 12, wherein the means for pseudorandomly distributing comprises:

means for calculating a remainder modulo P of a polynomial x^i, where i is an intended bit location in the second plurality of bits and P is a cyclic redundancy check polynomial derived from the third plurality of bits; and

means, for each bit of the first plurality of bits that is equal to one, for performing an exclusive-OR operation, bit by bit, of the third plurality of bits and the calculated remainder.

23. A generator configured to generate a message authentication code, comprising:

a processor configured to pseudorandomly distribute in a key-dependent manner a first plurality of message bits into a second plurality of bits;

a generator coupled to the distributor and configured to generate a third plurality of bits that comprises a cyclic redundancy check of the second plurality of bits; and

a transmitter coupled to the generator and configured to transmit the first plurality of message bits and a message authentication code that comprises the third plurality of bits.

24. The generator of claim 23, wherein the generator comprises lookup tables accessible by the microprocessor, and a set of software instructions stored in a memory element and executable by the microprocessor.

25. The generator of claim 23, wherein the generator comprises a shift register.

26. The generator of claim 23, wherein the processor is further configured to place a first bit of the first plurality of bits into an offset bit location in the second plurality of bits, and to skip an unpredictable number of bit locations in the second plurality of bits from the offset bit location.

27. The generator of claim 26, wherein the processor is further configured to put a next bit of the first plurality of bits into an initial bit location of the second plurality of bits.

28. The generator of claim 23, wherein the processor is further configured to divide the second plurality of bits into blocks of approximately uniform block size, and to place the first plurality of bits into random bit locations within each block, with one bit being placed into each block.

29. The generator of claim 23, wherein the processor comprises a calculator configured to calculate a remainder modulo P of a polynomial x^l,

where i is an intended bit location in the second plurality of bits and P is a cyclic redundancy check polynomial derived from the third plurality of bits, and a polynomial adder configured to perform, for each bit of the first plurality of bits that is equal to one, an exclusive-OR operation, bit by bit, of the third plurality of bits and the calculated remainder.

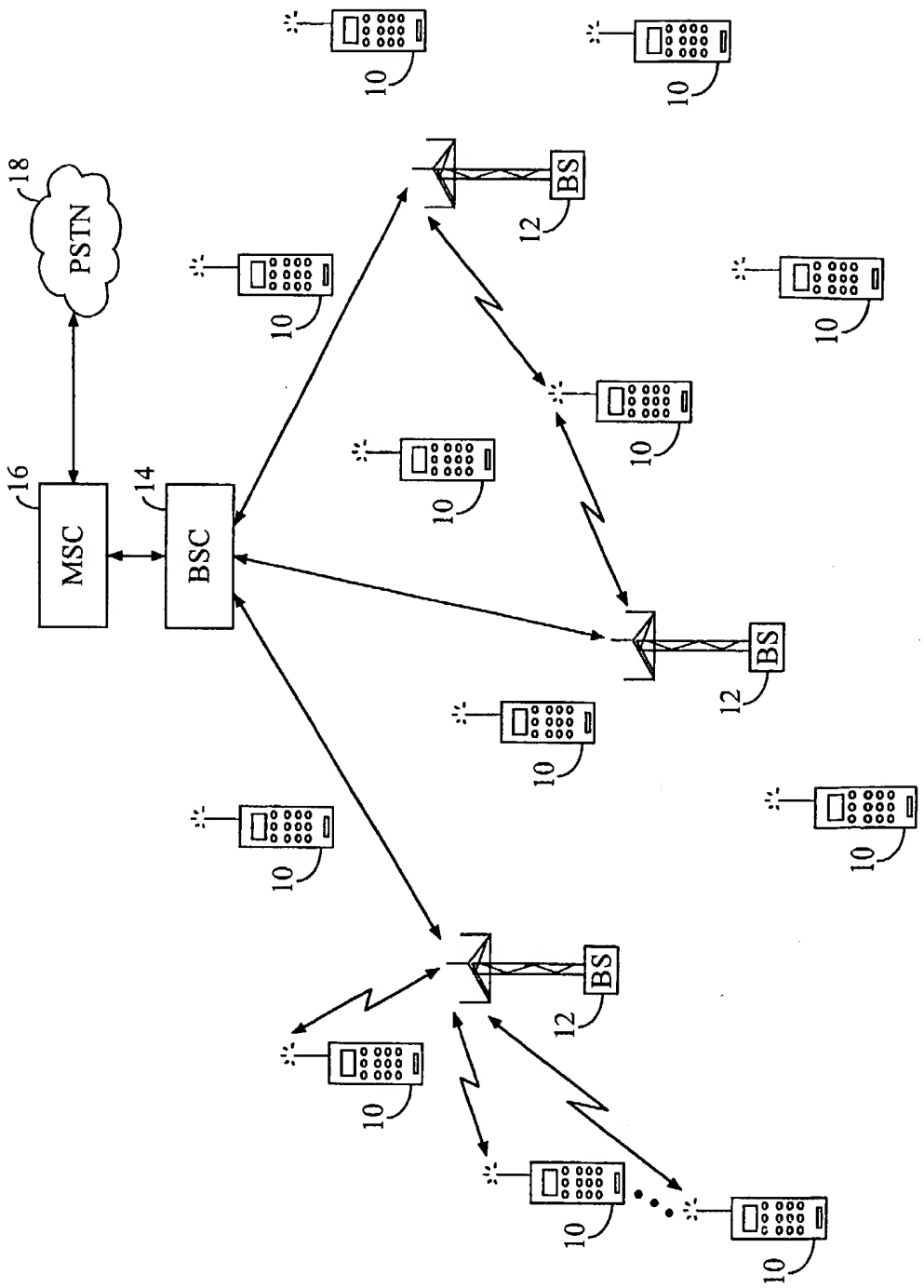


FIG. 1

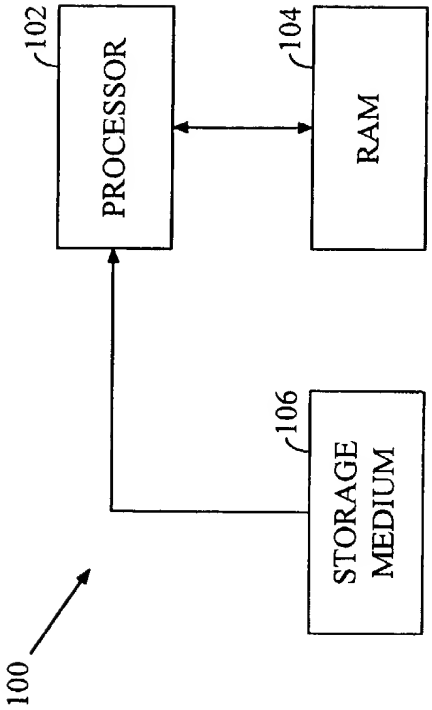


FIG. 2

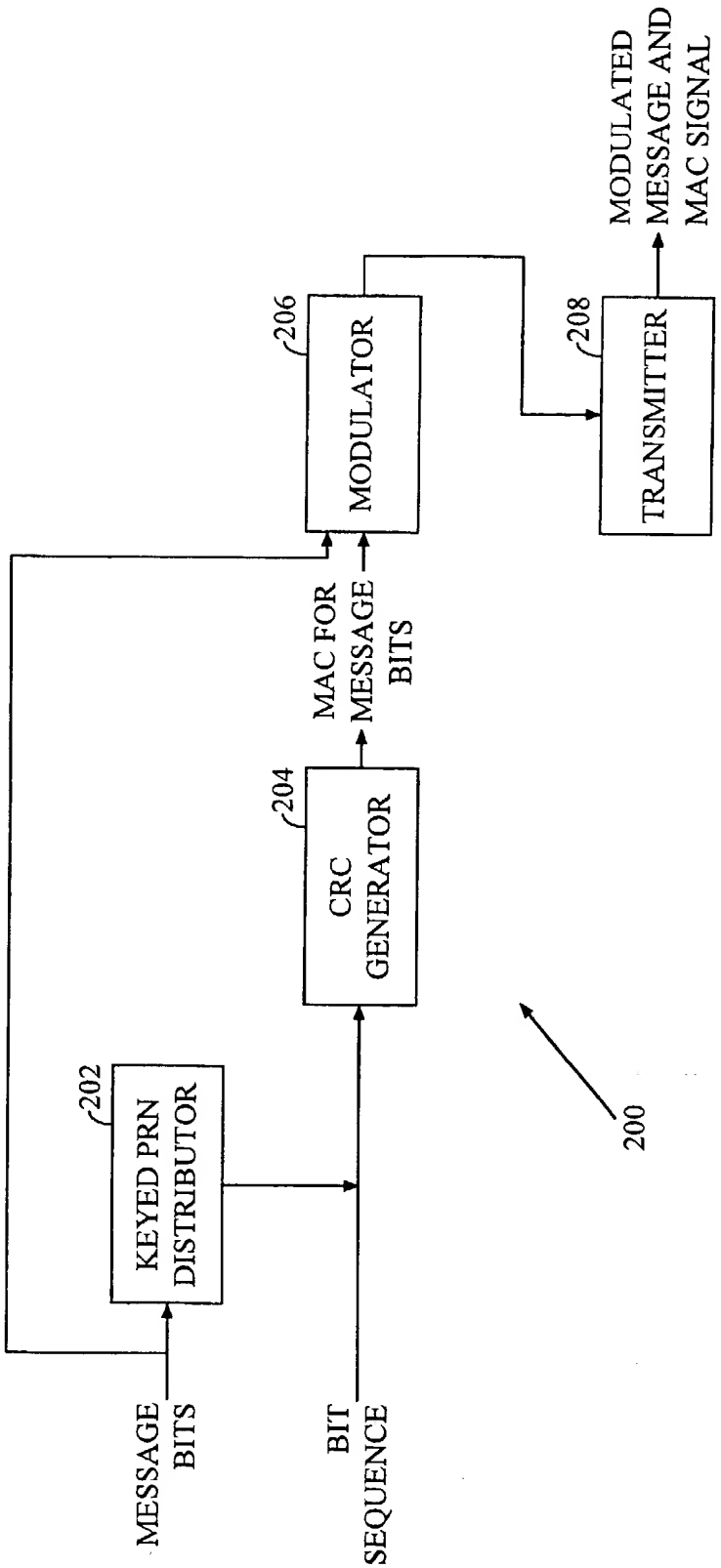

MODULATED MESSAGE AND MAC SIGNAL
206
MODULATOR
208
TRANSMITTER
MAC FOR MESSAGE BITS
204
CRC GENERATOR
202
KEYED PRN DISTRIBUTOR
200
MESSAGE BITS
BIT SEQUENCE

FIG. 3

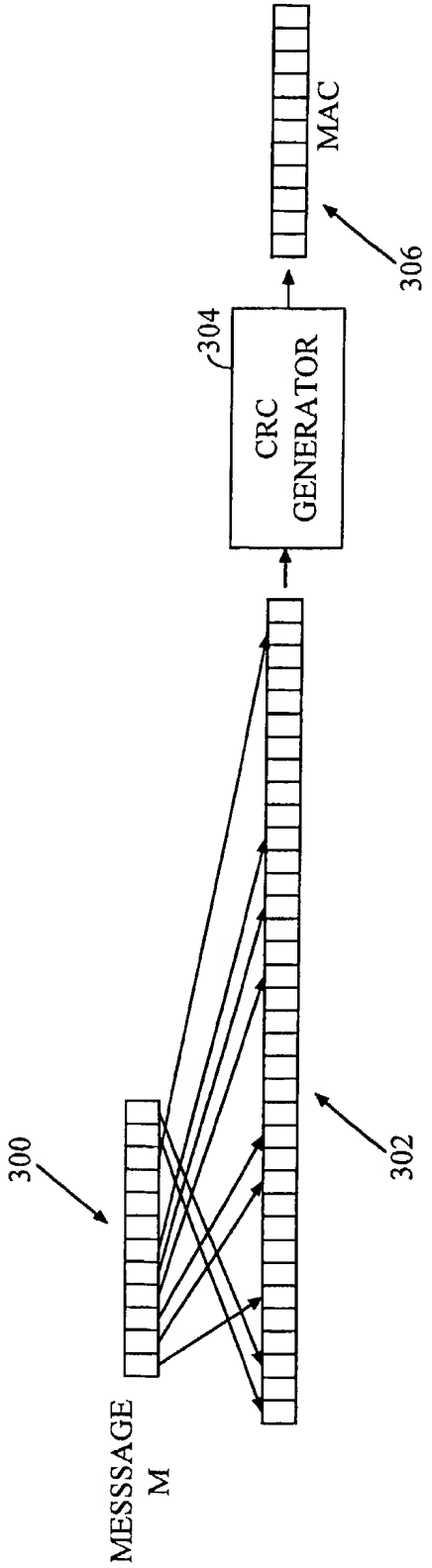


FIG. 4

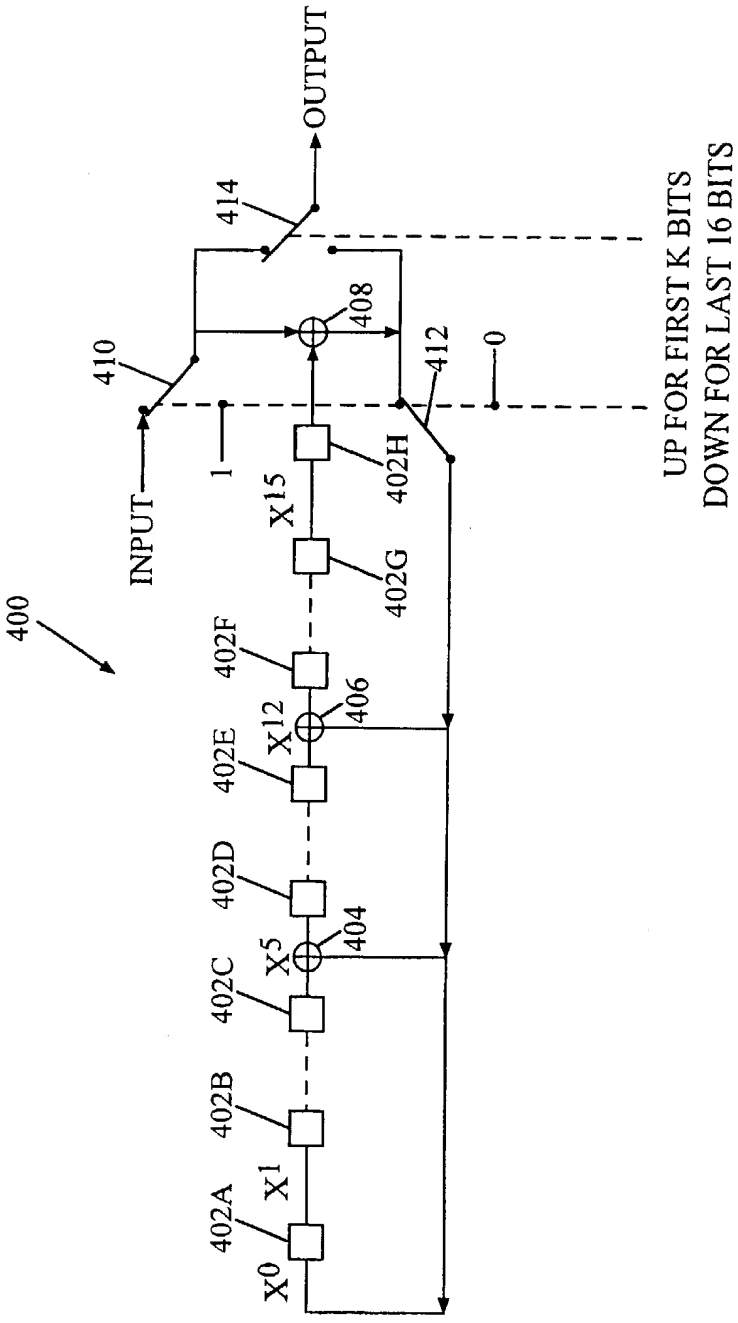

400
INPUT
410
1
414
OUTPUT
408
412
0
UP FOR FIRST K BITS
DOWN FOR LAST 16 BITS
402H
402G
402F
402E
402D
402C
402B
402A
X15
X12
X5
X1
X0
406
404

FIG. 5

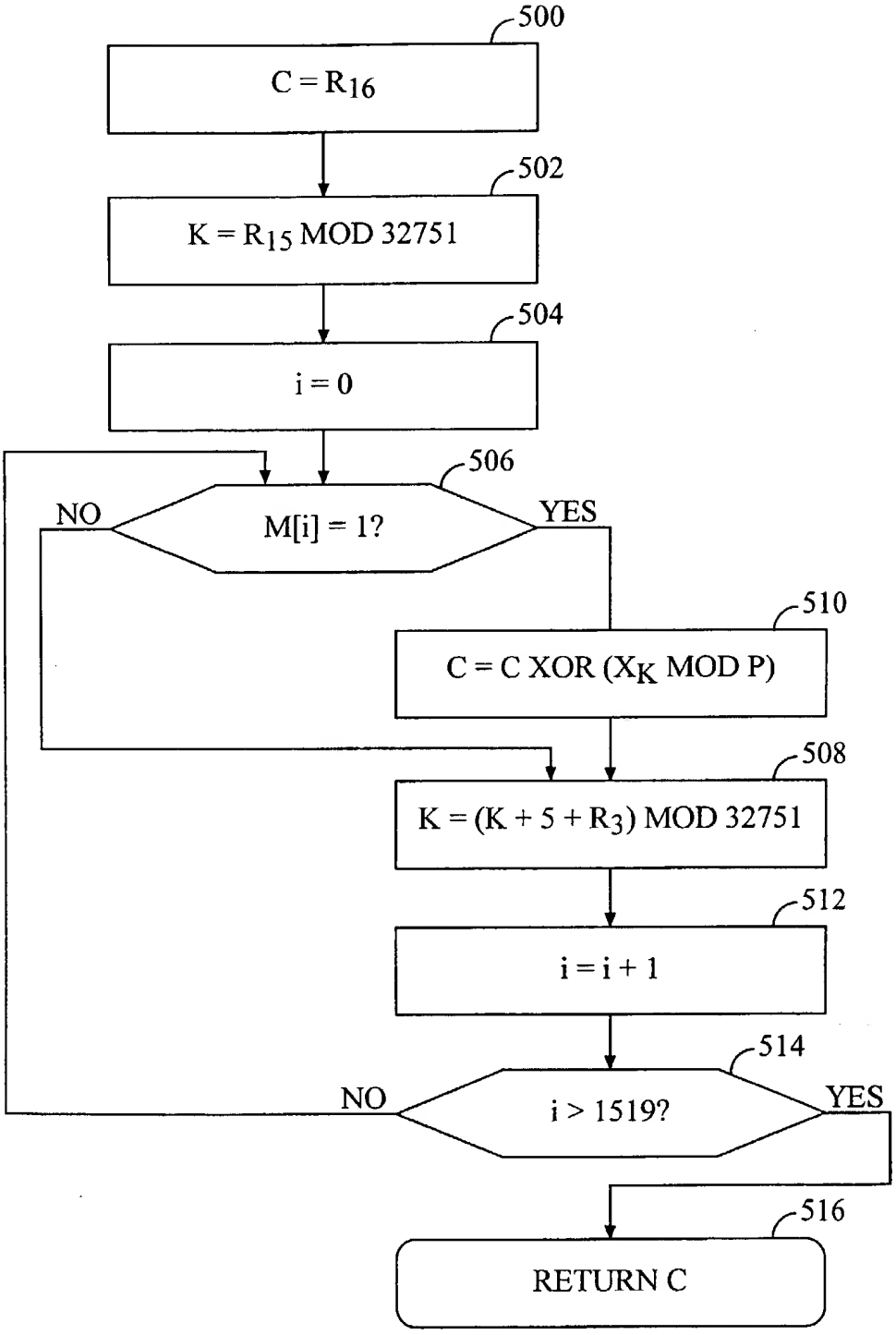

500
C = R16
502
K = R15 MOD 32751
504
i = 0
506
NO
M[i] = 1?
YES
510
C = C XOR (XK MOD P)
508
K = (K + 5 + R3) MOD 32751
512
i = i + 1
514
NO
i > 1519?
YES
516
RETURN C

FIG. 6

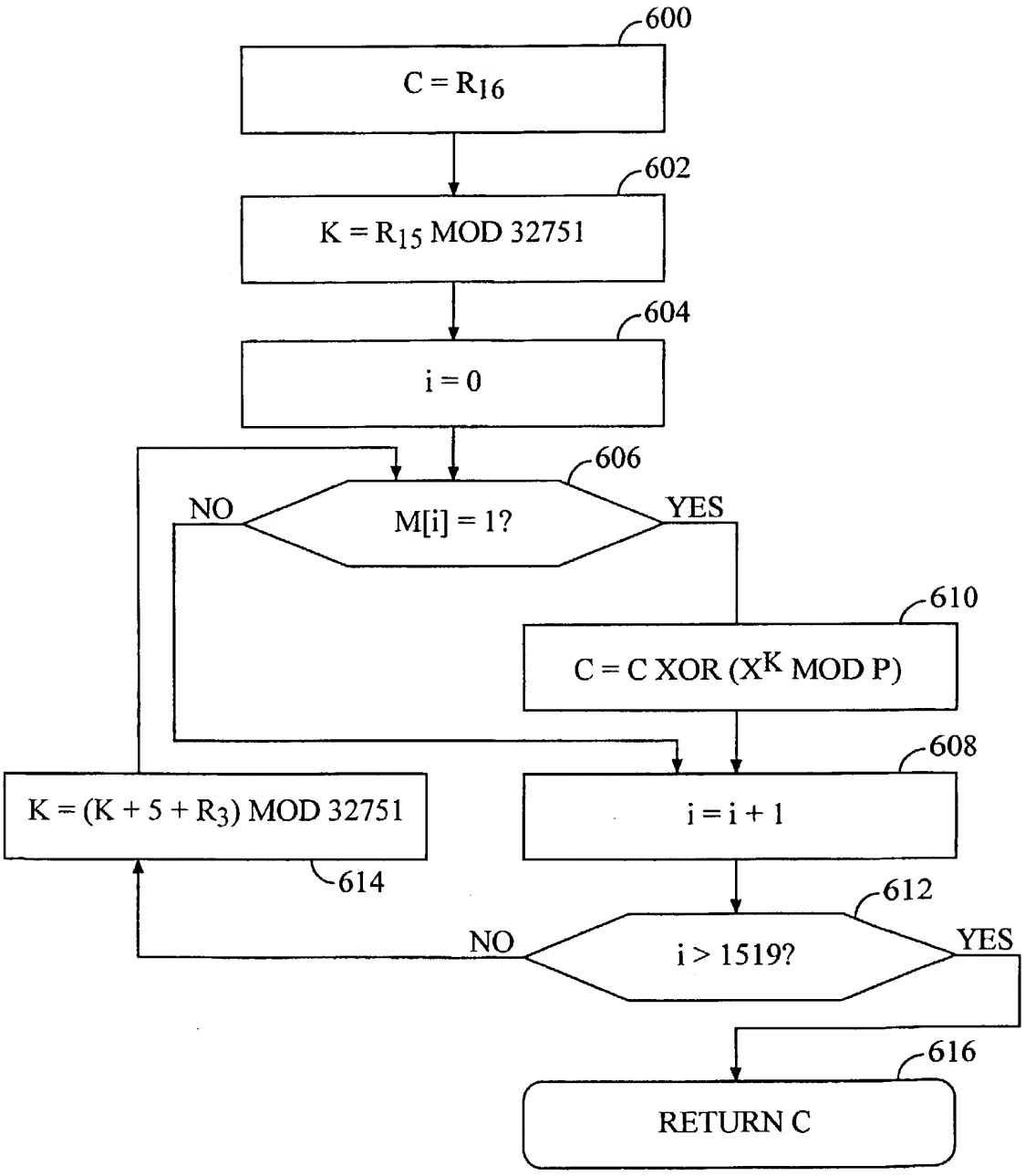

600
C = R16
602
K = R15 MOD 32751
604
i = 0
606
M[i] = 1?
NO
YES
610
C = C XOR (X^K MOD P)
608
i = i + 1
612
i > 1519?
NO
YES
614
K = (K + 5 + R3) MOD 32751
616
RETURN C

FIG. 7